Durante esta aula, vimos como a orientação a objetos é implementada no JavaScript sem a utilização da sintaxe de classes, e demos atenção especial ao `this` e ao conceito de cadeia de protótipo.

Baseado nisso, podemos afirmar que:

A sintaxe de classe não existia no JavaScript até o ES6, e foi implementada como "açúcar sintático" por cima do modelo de protótipo nativo do JavaScript. Isso aconteceu por, entre outros fatores, uma demanda da comunidade dev que já estava acostumada a utilizar classes em orientação a objetos e preferia essa forma a ter que utilizar os protótipos.

Alternativa correta! Embora hoje em dia seja possível programar em JavaScript utilizando apenas a sintaxe de classe, é importante saber o que acontece "por baixo dos panos", pois alguns aspectos da cadeia de protótipo (como o próprio objeto `prototype`) muitas vezes acabam aparecendo durante o dia a dia do desenvolvimento, especialmente quando temos que resolver bugs.

A palavra-chave `this` se refere ao contexto em que uma função está sendo executada; esse contexto só é determinado no momento da chamada da função e só é possível saber qual será o valor de `this` para uma função se soubermos em que contexto esta função será executada - em outras palavras, a que objeto ela fará referência.

Alternativa correta! O `this` é uma das palavras-chave mais importantes para se compreender como os objetos funcionam no javaScript, independente de ser um projeto orientado a objetos ou não. Para trabalharmos com `this` sempre precisamos ter em mente qual o **contexto** em que uma função é executada.

No modelo de protótipo, os objetos "herdam" recursos uns dos outros através da chamada propriedade protótipo. Quando criamos um objeto e definimos seu protótipo através de `Object.setPrototypeOf(objetoQueHerda, objetoBase)`, estabelecemos uma **cadeia de protótipos** que começa no próprio tipo `Object` e vai até o último nível de objeto criado através dessa cadeia.

Alternativa correta! A cadeia de protótipos é a base original de programação orientada a objetos com JavaScript. Praticamente tudo, em JavaScript, é considerado um objeto, e estes objetos recebem um protótipo, de onde vêm as propriedades e métodos que todos os objetos do mesmo tipo compartilham.

D

Vimos como criar objetos a partir de "modelos" com funções construtoras e `Object.create()`. Porém, após a implementação das classes, estas duas formas estão em processo de descontinuação (*deprecation*).

A criação de objetos com `Object.create()` é a forma mais indicada para trabalhar com o modelo de protótipo. As funções construtoras com `new` estão mais próximas, sintaticamente falando, do conceito de classes. Embora hoje em dia o uso de classes esteja bastante consolidado, a programação com protótipos e as funções construtoras não estão depreciadas e ainda são utilizadas.